PCT/DE03/03453
10/532732

BUNDESREPUBLIK DEUTSCHLAND

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: 102 50 641.8

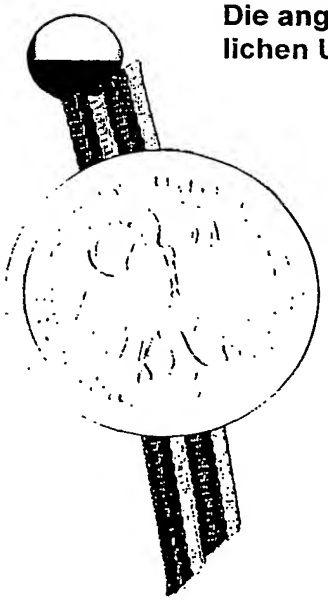
Anmeldetag: 30. Oktober 2002

Anmelder/Inhaber: Siemens Aktiengesellschaft, München/DE

Bezeichnung: Auf- und abwärtskompatible Schemaevolution

IPC: G 06 F 17/30

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.



München, den 30. Oktober 2003
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Schäfer

A 9161
02/00
EDV-L

Beschreibung

Auf- und abwärtskompatible Schemaevolution

Die Erfindung betrifft ein Verfahren sowie ein System zur Definition von Strukturen von Objekt- und/oder Datenmodellen, mit mindestens einem Schema zur Beschreibung der Strukturen.

Strukturen von Objekt- und Datenmodellen werden bei der Softwareentwicklung typischer Weise mit Klassen-/Typmodellen und Schemas (z. B. Datenbankschemas, XML-Schemas) definiert (XML = Extensible Markup Language). Im Folgenden soll unter dem Begriff Schema auch Klassen-/Typmodell verstanden werden. Schemas dienen also zur Beschreibung, wie Daten abgelegt werden. Die abzulegenden Daten ändern sich im Allgemeinen über die Zeit in ihrer Struktur. Daher ist es notwendig, auch das jeweils zugrundeliegende Schema zu ändern, d. h. es findet eine Schemaevolution statt. Folgende Dinge sind bei dieser Schemaevolution wesentlich: Zum einen muss die Version eines Schemas identifiziert werden können. Zum anderen sollte die Kompatibilität zwischen verschiedenen Schemata geklärt und angegeben werden können. Kompatibilität zwischen zwei Schemas bedeutet hier, dass Daten, die bzgl. dem einen Schema korrekt abgelegt sind, auch bzgl. des anderen Schemas korrekt sind. Unter "Daten sind zu einem Schema korrekt" ist zu verstehen, dass die Daten inhaltlich korrekt von einer Applikation interpretiert werden können, wenn der Applikation die Bedeutung der Strukturen aus dem Schema bekannt ist. Mit der Kurzform "Daten eines Schemas" seien Daten bezeichnet, die bzgl. eines Schemas korrekt sind. Bei der Kompatibilität von Schemas unterscheidet man üblicherweise zwischen Aufwärtskompatibilität (= Daten eines alten Schemas sind korrekt bzgl. eines neuen Schemas) und Abwärtskompatibilität (= Daten eines neuen Schemas, die zu Strukturen des alten Schemas inhaltlich korrespondieren, sind korrekt bzgl. eines alten Schemas). Die Eigenschaften Aufwärtskompatibilität und Abwärtskompatibilität zwischen den Schemaversionen sind

eminent wichtig, da sie direkte Auswirkungen auf die Machbarkeit und Aufwände für die Migration von Endanwenderdaten von Softwareprodukten haben. Für die Schema Implementierung wird heutzutage oft der W3C-Standard XML-Schema (W3C = World Wide Web Consortium) eingesetzt. Dort gibt es Mechanismen für die Schemaevolution. Setzt man diese Mechanismen so ein, wie es standardmäßig in der objektorientierten Softwareentwicklung üblich ist, so erhält man aufwärtskompatible XML-Schemas, die jedoch nicht abwärtskompatibel sind.

Der Erfindung liegt die Aufgabe zugrunde, eine auf- und abwärtskompatible Schemaevolution zu ermöglichen.

Diese Aufgabe wird durch ein Verfahren zur Definition von Strukturen von Objekt- und/oder Datenmodellen gelöst, bei welchem Schemata die Strukturen beschreiben, wobei in einem ersten Attribut eines Schemas eine Kennzeichnung einer Version des jeweiligen Schemas erfolgt, wobei der im jeweiligen Schema verwendete Namensraum und die im jeweiligen Schema verwendeten Typ- und Elementnamen unabhängig von der Version beibehalten werden, wobei Typen und Elemente nur unter Beibehaltung des Typ- bzw. Elementnamens erweitert werden und wobei in Schemata einer neueren Version nicht erweiterte Typen und Elemente unverändert von den jeweiligen in Schemata einer älteren Version verwendeten Typen bzw. Elementen übernommen werden.

Diese Aufgabe wird durch ein System zur Definition von Strukturen von Objekt- und/oder Datenmodellen gelöst, mit mindestens einem Schema zur Beschreibung der Strukturen, wobei ein erstes Attribut eines Schemas zur Kennzeichnung einer Version des jeweiligen Schemas vorgesehen ist, wobei der im jeweiligen Schema verwendete Namensraum und die im jeweiligen Schema verwendeten Typ- und Elementnamen unabhängig von der Version beibehalten werden, wobei ein Mechanismus zur Erweiterung der Typen und Elemente unter

Beibehaltung des Typ- bzw. Elementnamens und zur unveränderten Übernahme von in Schemata einer älteren Version verwendeten, nicht erweiterten Typen bzw. Elementen in Schemata einer neueren Version vorgesehen ist.

Durch die vorliegende Erfindung wird ein Weg aufgezeigt, eine Schemaevolution so durchzuführen, dass die Schemas sowohl aufwärts- als auch abwärtskompatibel sind. Die Erfindung ermöglicht eine Schemaevolution, ohne die Namen der Daten zu ändern. Grundidee dabei ist, den Namensraum, die Typ- und Elementnamen beim Übergang auf eine neue Schemaversion beizubehalten und eine Schemaversionskennung zu benutzen. Ein Namensraum ist eine Sammlung von Namen, die durch einen eindeutigen Bezeichner identifiziert werden. Ein Namensraum ist damit so etwas wie ein Container für Elemente und Attribute, der selbst einen einmaligen Namen besitzt. Ein Namensraum wird auch als „Namespace" bezeichnet.

Die Versionierung der Schemas wird ausschließlich über Attribute abgebildet. Dabei wird ein erstes Attribut eines Schemas zur Kennzeichnung einer Version des jeweiligen Schemas benutzt. Gemäß einer vorteilhaften Ausgestaltung der Erfindung kann ein Kalenderdatum über ein zweites Attribut einer Version eines Schemas zugeordnet werden. Das Kalenderdatum der jeweiligen Schemaversion kann z. B. in den sogenannten "Annotations" zum Schema über ein Attribut "versiondate" abgelegt werden.

Werden die Schemata durch eine erweiterbare Auszeichnungssprache, z. B. XML, beschrieben, so erreicht man neben Einheitlichkeit und Erweiterbarkeit auch systematische Validierbarkeit.

Nachfolgend wird die Erfindung anhand des in der Figur dargestellten Ausführungsbeispiels näher beschrieben und erläutert.

Die Figur zeigt ein System zur Definition von Strukturen von Objekt- und/oder Datenmodellen, mit Schemata zur Beschreibung der Strukturen.

Im Ausführungsbeispiel dargestellt sind ein erstes Schema XS1 älterer Version und ein zweites Schema XS2 neuerer Version, welche beide die Strukturen eines Objektmodells OM beschreiben. Der Pfeil 30 symbolisiert die Schemaevolution. Die Schemata XS1, XS2 enthalten Typen und Elemente 11:.14, 21..24, welchen Typ- bzw. Elementnamen 11a..14a, 21a..24a zugeordnet sind. Den Schemata XS1, XS2 ist ein Namensraum 1 zugeordnet. In ersten Attributen 10, 20 und zweiten Attributen D1, D2 der Schemata XS1, XS2 können Versionkennungen bzw. Kalenderdaten hinterlegt werden.

Im Folgenden wird die Erfindungsidee anhand des für die Schema-Implementierung oft eingesetzten W3C-Standard XML-Schema erläutert. Die beschriebenen Mechanismen lassen sich jedoch prinzipiell bei der Beschreibung der Strukturen beliebiger Objekt- und Datenmodelle einsetzen, unabhängig von Standards. Bisher wurden zwar aufwärtskompatible XML-Schemas definiert, aber keine abwärtskompatiblen XML-Schemas. Die Definition aufwärtskompatibler XML-Schemas wurde folgendermaßen gelöst: Bei XML-Schema gibt es die Mechanismen "Typableitung", Targetnamespaces und das Attribut "version" beim Element <xsd:schema>. Setzt man diese Mechanismen so ein, wie es in der objektorientierten Softwareentwicklung bekannt ist, bedeutet das, dass abgeleitete Typen andere Namen erhalten als ihre Vatertypen. Dadurch erhalten Daten von neueren XML-Schemas andere Namen als die entsprechenden Daten von alten XML-Schemas. Über die Typableitungsbeziehung sind neuen Applikationen sowohl die alten als auch die neuen Namen von Daten bekannt. Sie können deshalb alte und neue Daten interpretieren. Die XML-Schemas sind also aufwärtskompatibel. Da sich jedoch Namen von Daten geändert haben, können alte Applikationen, denen die neuen Namen nicht bekannt sein können, neue Daten nicht interpretieren. Die

Schemas sind nicht abwärtskompatibel. Versionsübergreifend XML-Dokumente zu lesen ist dann nur noch möglich, indem man Konverter einsetzt, welche die XML-Dokumente in die jeweils richtige Version konvertieren. Dies hat aber den entscheidenden Nachteil, dass man mit neuen Daten alleine nichts anfangen kann. Man braucht dann stets einen passenden Konverter.

Für den Übergang von einer XML-Schema-Version zur nächsten stellt XML-Schema verschiedene Mittel zur Verfügung. Um Elementdefinitionen erweitern zu können, ohne den Namen eines Elementes zu ändern, bietet XML-Schema das Mittel der Redefinition von Elementtypen. Die Idee einer Redefinition ist es, eine "Vererbung" durchzuführen, ohne den Namen des Elementtyps zu ändern. Der Mechanismus der Redefinition beinhaltet auch die Übernahme nicht redefinierter Typen aus der alten Schemadefinition. D. h. durch die Benutzung der Redefinition wird gleichzeitig ein "Include-Mechanismus" zur Übernahme von alten Typen ausgelöst. Dies unterstützt auch eine aufwärtskompatible Weiterentwicklung eines Schemas. Anhand eines schematischen Beispiels wird im Folgenden die Umsetzung eines Übergangs von einer XML- Schemaversion zur nächsten beschrieben. Zu betrachten sind dazu die XML-Schemaversionen, die zugehörigen Namespaces und die Typdefinitionen im jeweiligen XML-Schema. Die Versionierung der Schemas wird ausschließlich über Attribute abgebildet. Dabei wird das Attribut "version" des Elementes "xsd:schema" von XML-Schema benutzt. Außerdem kann das Datum der Schemaversion z. B. in den "Annotations" zum Schema über ein Attribute "versiondate" abgelegt werden.

Die Typen „Project", „HW", „Comm" sind nur beispielhaft und stehen für beliebige Typen. Alle drei Typen sind sowohl in de Version 1.0 als auch in der Version 2.0 vorhanden. Die Typen „HW" und „Comm" bleiben unverändert. Der Typ „Project" wird über Redefinition in Version 2.0 verändert. In der Version 2.0 wird zusätzlich der Typ Monitoring neu definiert. Es

wurde für eine neue Schemaversion kein neuer Namespace eingeführt. Außerdem wurden die lokalen Namen der Typen beibehalten. Damit haben sich insgesamt die Namen der bereits in Version 1.0 vorhandenen Typen nicht geändert. Die zu einem neuen Schema korrekten Daten, die inhaltlich zu Strukturen des alten Schemas korrespondieren, sind auch bzgl. des alten Schemas korrekt. Die Schemaevolution ist abwärtskompatibel. Da das neue Schema durch "Ableitung" aus dem alten Schema hervorgegangen ist, ist die Schemaevolution auch aufwärtskompatibel. Damit ist die Schemaevolution aufwärts- und abwärtskompatibel. Außerdem gilt im originären Sinne der "Validierbarkeit" des W3C, dass die alten und die neuen Daten (XML-Dokumente) bzgl. des neuen Schemas gültig sind. Das Versionierungskonzept des Standards XML-Schema und der Redefinition-Mechanismus von XML-Schema werden also eingesetzt um die Aufwärts- und Abwärtskompatibilität von Daten bei der Schemaevolution zu erreichen. Es sei betont, das sich die hier verwendete Definition der Aussage "Daten (und damit XML-Dokumente) sind korrekt" bzgl. eines XML-Schemas unterscheidet von der Definition des W3C: "Daten (XML-Dokumente) sind gültig bzgl. eines XML-Schemas". Während die Definition des W3C rein syntaktischer Natur ist, d. h. den Aufbau eines XML-Dokumentes betrifft, ist die hier verwendete Definition über den semantischen Inhalt und die Interpretierbarkeit der Daten bestimmt.

Das obige schematische Beispiel sieht in XML- und XML-Schema-Syntax folgendermaßen aus:

XML-Instanzdokument zu Schema Version 1.0

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="Namespace1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="Namespace1 D1.xsd">
   <Project>
      <HW/>
   </Project>
```

```
   <Project>
      <HW/>
   </Project>
</Document>
```

XML-Schema zu Version 1.0: Datei D1.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="Namespace1" xmlns:ns1="Namespace1"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1.0">
<xs:annotation>
   <xs:documentation>This schema is the first version of this schema
type</xs:documentation>
   <xs:appinfo>
      <prim:schemainfo versiondate="20011206" />
   </xs:appinfo>
</xs:annotation>

   <xs:element name="Document" type="ns1:DocumentT">
      <xs:annotation>
         <xs:documentation>Comment describing your root element</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="DocumentT">
      <xs:sequence>
         <xs:element ref="ns1:Project" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:element name="Project" type="ns1:ProjectT"/>
   <xs:complexType name="ProjectT">
      <xs:sequence>
         <xs:element ref="ns1:HW"/>
      </xs:sequence>
   </xs:complexType>
      <xs:element name="HW" type="ns1:HWT"/>
   <xs:complexType name="HWT"/>
```

```
   <xs:element name="Comm" type="ns1:CommT"/>
   <xs:complexType name="CommT"/>
   </xs:schema>
```

XML-Instanzdokument zu Schema Version 2.0

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="Namespace1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="Namespace1
D2.xsd">
   <Project>    <!-- Elementtype redefined in Schema version 2.0 -->
      <HW/>       <!-- Element already existing as subelement of Project in Schema
version 1.0 -->
      <Comm/>     <!-- Reuse of Element defined in Schema version 1.0 -->
      <Monitoring/>  <!-- Element newly defined in Schema version 2.0 -->
   </Project>
   <Project>
      <HW/>
      <Comm/>
      <Monitoring/>
   </Project>
</Document>
```

XML-Schema zu Version 2.0: Datei D2.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="Namespace1" xmlns="Namespace1"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="2.0">
<xs.annotation>
   <xs:documentation>This schema enhances the previous schema
version</xs:documentation>
   <xs:appinfo>
      <prim:schemainfo versiondate="20011218" />
   </xs:appinfo>
</xs:annotation>
```

```
<xs:redefine schemaLocation="D1.xsd">
   <xs:complexType name="ProjectT">
      <xs:complexContent>
         <xs:extension base="ProjectT">
            <xs:sequence>
               <xs:element ref="Comm"/>
               <xs:element ref="Monitoring"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:redefine>
<xs:element name="Monitoring" type="MonitoringT"/>
<xs:complexType name="MonitoringT"/>
</xs:schema>
```

Zusammenfassend betrifft die Erfindung somit ein Verfahren sowie ein System zur Definition von Strukturen von Objekt- und/oder Datenmodellen OM, mit mindestens einem Schema XS1, XS2 zur Beschreibung der Strukturen. Eine auf- und abwärtskompatible Schemaevolution wird dadurch erreicht, dass in einem ersten Attribut 10, 20 eines Schemas XS1, XS2 eine Kennzeichnung einer Version des jeweiligen Schemas XS1, XS2 erfolgt, wobei der im jeweiligen Schema XS1, XS2 verwendete Namensraum 1 und die im jeweiligen Schema XS1, XS2 verwendeten Typ- und Elementnamen 11a..14a, 21a..24a unabhängig von der Version beibehalten werden, wobei Typen und Elemente 11..14, 21..24 nur unter Beibehaltung des Typ- bzw. Elementnamens 11a..14a, 21a..24a erweitert werden und wobei in Schemata XS2 einer neueren Version nicht erweiterte Typen und Elemente 21..24 unverändert von den jeweiligen in Schemata XS1 einer älteren Version verwendeten Typen bzw. Elementen 11..14 übernommen werden.

Patentansprüche

1. Verfahren zur Definition von Strukturen von Objekt- und/oder Datenmodellen (OM), bei welchem Schemata (XS1, XS2) die Strukturen beschreiben, wobei in einem ersten Attribut (10, 20) eines Schemas (XS1, XS2) eine Kennzeichnung einer Version des jeweiligen Schemas (XS1, XS2) erfolgt, wobei der im jeweiligen Schema (XS1, XS2) verwendete Namensraum (1) und die im jeweiligen Schema (XS1, XS2) verwendeten Typ- und Elementnamen (11a..14a, 21a..24a) unabhängig von der Version beibehalten werden, wobei Typen und Elemente (11..14, 21..24) nur unter Beibehaltung des Typ- bzw. Elementnamens (11a..14a, 21a..24a) erweitert werden und wobei in Schemata (XS2) einer neueren Version nicht erweiterte Typen und Elemente (21..24) unverändert von den jeweiligen in Schemata (XS1) einer älteren Version verwendeten Typen bzw. Elementen (11..14) übernommen werden.

2. Verfahren nach Anspruch 1,
d a d u r c h g e k e n n z e i c h n e t ,
dass ein Kalenderdatum über ein zweites Attribut (D1, D2) einer Version eines Schemas (XS1, XS2) zugeordnet werden kann.

3. Verfahren nach Anspruch 1 oder 2,
d a d u r c h g e k e n n z e i c h n e t ,
dass die Schemata (XS1, XS2) durch eine erweiterbare Auszeichnungssprache beschrieben werden.

4. System zur Definition von Strukturen von Objekt- und/oder Datenmodellen (OM), mit mindestens einem Schema (XS1, XS2) zur Beschreibung der Strukturen, wobei ein erstes Attribut (10, 20) eines Schemas (XS1, XS2) zur Kennzeichnung einer Version des jeweiligen Schemas (XS1, XS2) vorgesehen ist, wobei der im jeweiligen Schema (XS1, XS2) verwendete Namensraum (1) und die im jeweiligen Schema (XS1, XS2) verwendeten Typ- und Elementnamen (11a..14a, 21a..24a)

unabhängig von der Version beibehalten werden, wobei ein Mechanismus zur Erweiterung der Typen und Elemente (11..14, 21..24) unter Beibehaltung des Typ- bzw. Elementnamens (11a..14a, 21a..24a) und zur unveränderten Übernahme von in Schemata (XS1) einer älteren Version verwendeten, nicht erweiterten Typen bzw. Elementen (11..14, 21..24) in Schemata (XS2) einer neueren Version vorgesehen ist.

5. System nach Anspruch 4,
d a d u r c h g e k e n n z e i c h n e t ,
dass ein Kalenderdatum über ein zweites Attribut (D1, D2) einer Version eines Schemas (XS1, XS2) zugeordnet wird.

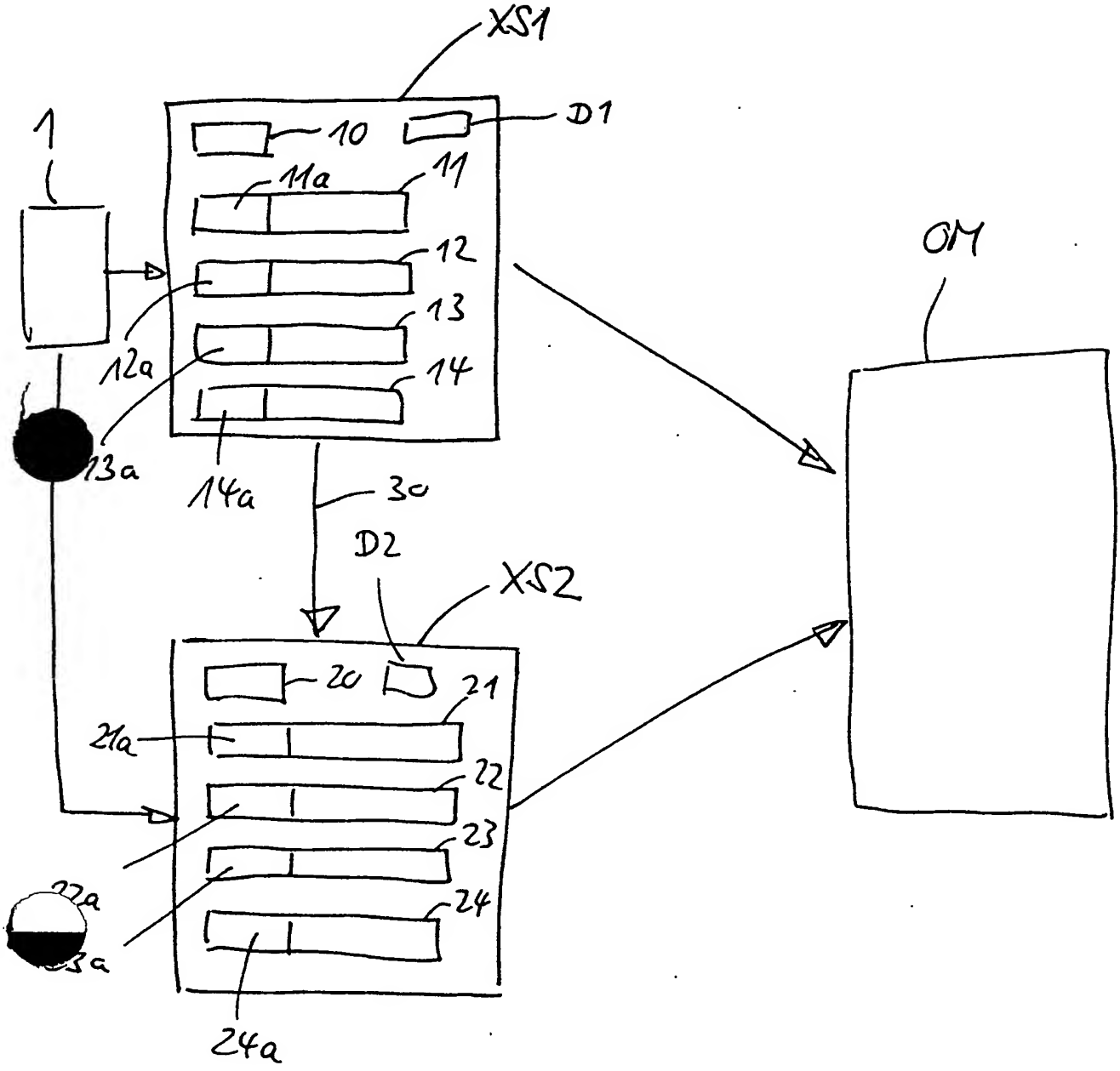
6. System nach Anspruch 4 oder 5,
d a d u r c h g e k e n n z e i c h n e t ,
dass die Schemata (XS1, XS2) durch eine erweiterbare Auszeichnungssprache beschrieben sind.

Zusammenfassung

Auf- und abwärtskompatible Schemaevolution

Die Erfindung betrifft ein Verfahren sowie ein System zur Definition von Strukturen von Objekt- und/oder Datenmodellen (OM), mit mindestens einem Schema (XS1, XS2) zur Beschreibung der Strukturen. Eine auf- und abwärtskompatible Schemaevolution wird dadurch erreicht, dass in einem ersten Attribut (10, 20) eines Schemas (XS1, XS2) eine Kennzeichnung einer Version des jeweiligen Schemas (XS1, XS2) erfolgt, wobei der im jeweiligen Schema (XS1, XS2) verwendete Namensraum (1) und die im jeweiligen Schema (XS1, XS2) verwendeten Typ- und Elementnamen (11a..14a, 21a..24a) unabhängig von der Version beibehalten werden, wobei Typen und Elemente (11..14, 21..24) nur unter Beibehaltung des Typ- bzw. Elementnamens (11a..14a, 21a..24a) erweitert werden und wobei in Schemata (XS2) einer neueren Version nicht erweiterte Typen und Elemente (21..24) unverändert von den jeweiligen in Schemata (XS1) einer älteren Version verwendeten Typen bzw. Elementen (11..14) übernommen werden.

FIG


XS1
D1
1
10
11a
11
12
13
12a
14
13a
14a
30
OM
D2
XS2
20
21
21a
22
23
22a
24
24a

FIG